Pauly, Inn - Baseball Final Project

May 15, 2023

[62]:
```
import warnings
warnings.filterwarnings('ignore')


import pandas as pd
import numpy as np
from plotnine import *
# import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import accuracy_score, confusion_matrix,\
 f1_score, recall_score, precision_score, roc_auc_score,␣
 ↪ConfusionMatrixDisplay, r2_score, mean_squared_error, silhouette_score
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans

from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.decomposition import PCA

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split # simple TT split cv
import seaborn as sb
```

#CPSC-392 Final Project - Baseball Statistics #####Jack Pauly & Zachary Inn

##Question 1: Barry Bonds *Barry Bonds holds the pro-baseball record for most career home runs, however after the 1998 season, he started using illegal performance enhancing drugs (PEDs). How much of an impact did PEDs have on Barry Bonds' career stat totals? How would Barry Bonds' WAR and HR total look like if he had played for the same amount of time, but did not use PEDs?*

- Variables Involved: OBP (Continuous), BA (Ratio), OBS (Continuous), WAR (Continuous), SLG (Continuous), Walks (BB) (Interval), Doubles (2B) (Interval), Home Runs (HR) (Interval), and PEDs (binary)
- Background: The best metric to measure a batter's worth is OPS ("On-base plus slugging"), a sum of the batter's OBP ("On Base Percentage") and SLG ("slugging"). OBP measures the batter's ability to get on base, whether that be via hits, walks, etc. SLG measures the batter's ability to hit for power by weighing the different types of hits a batter can get (singles, doubles, triples, and home runs). We will be predicting Barry Bonds' OPS had he not used

steroids.

- Cleaning: It's generally agreed upon that Barry Bonds started to use Performance Enhancing Drugs (PEDs) after the 1998 Season. Thus, we separate the data into rows before 1999 and after 1999.
- Modeling/Computation: Z-Score the continuous variables. Then, use an 80/20 train/test split on the Bonds' statistics from 1986 to 1999. Fit a linear regression model to predict WAR.
- Graphs: A line chart to show Bonds' values prior to the 1999 season, predicted statistics, and actual statistics.

[63]:
```
barry_bonds_df = pd.read_csv('https://raw.githubusercontent.com/Jackmpauly/
 ↪CPSC-392-Final-Project/main/csv/barry_bonds.csv')
barry_bonds_df.head()
```

[63]:
```
   Year  Age   Tm  PED  Lg    G   PA   AB    R    H  …  GDP  HBP  SH  SF  \
0  1986   21  PIT    0  NL  113  484  413   72   92  …    4    2   2   2
1  1987   22  PIT    0  NL  150  611  551   99  144  …    4    3   0   3
2  1988   23  PIT    0  NL  144  614  538   97  152  …    3    2   0   2
3  1989   24  PIT    0  NL  159  679  580   96  144  …    9    1   1   4
4  1990   25  PIT    0  NL  151  621  519  104  156  …    8    3   0   6

   IBB  WAR  WAA     Pos      Awards  Year-additional
0    2  3.5  1.9    *8/H       RoY-6            -9999
1    3  5.8  3.7  *78H/9         NaN            -9999
2   14  6.3  4.1   *7H/8         NaN            -9999
3   22  8.0  5.8    *7/H         NaN            -9999
4   15  9.7  7.8   *7/H8  ASMVP-1GGSS            -9999

[5 rows x 34 columns]
```

[64]:
```
# Setting up the predictors and doing a train_test_split
pre_PED_df = barry_bonds_df.loc[barry_bonds_df['PED'] == 0]
post_PED_df = barry_bonds_df.loc[barry_bonds_df['PED'] == 1]



predictors = ['OBP', 'BA', 'OPS', 'SLG', 'BB', '2B', 'HR']
X = pre_PED_df[predictors]
y = pre_PED_df['WAR']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

# Z-scoring X_train and X_test
z = StandardScaler()

z.fit(X_train)

X_train[predictors] = z.fit_transform(X_train[predictors])
X_test[predictors] = z.transform(X_test[predictors])
```

```
# fitting Linear Regression Model

lr = LinearRegression()
lr.fit(X_train, y_train)

# z-scoring post PED stats

post_PED_df[predictors] = z.fit_transform(post_PED_df[predictors])
pre_PED_df[predictors] = z.fit_transform(pre_PED_df[predictors])
```

[65]:
```
# Making the combined dataframe

combined_df = pd.DataFrame(barry_bonds_df[['Year', 'WAR', 'PED']])
predicted_non_PED_df = pd.DataFrame({'Year': post_PED_df['Year']})
predicted_non_PED_df['WAR'] = lr.predict(pre_PED_df[predictors])[-9:]
predicted_non_PED_df['PED'] = 0
combined_df = combined_df.append(predicted_non_PED_df)
```

[66]:
```
# Finding the overall slope of the linear regression

coefs = lr.coef_
r_squared = lr.score(X, y)

# calculate overall slope
overall_slope = np.average(coefs, weights=abs(coefs)/sum(abs(coefs)))
overall_slope
lr.intercept_
```

[66]:
```
7.680000000000104
```

[67]:
```
post_PED_df['WAR']
predicted_non_PED_df['WAR']
```

[67]:
```
13     6.786382
14     9.281927
15    10.098035
16     7.389249
17     5.157416
18     8.674757
19    11.707401
20    10.875000
21     8.976372
Name: WAR, dtype: float64
```

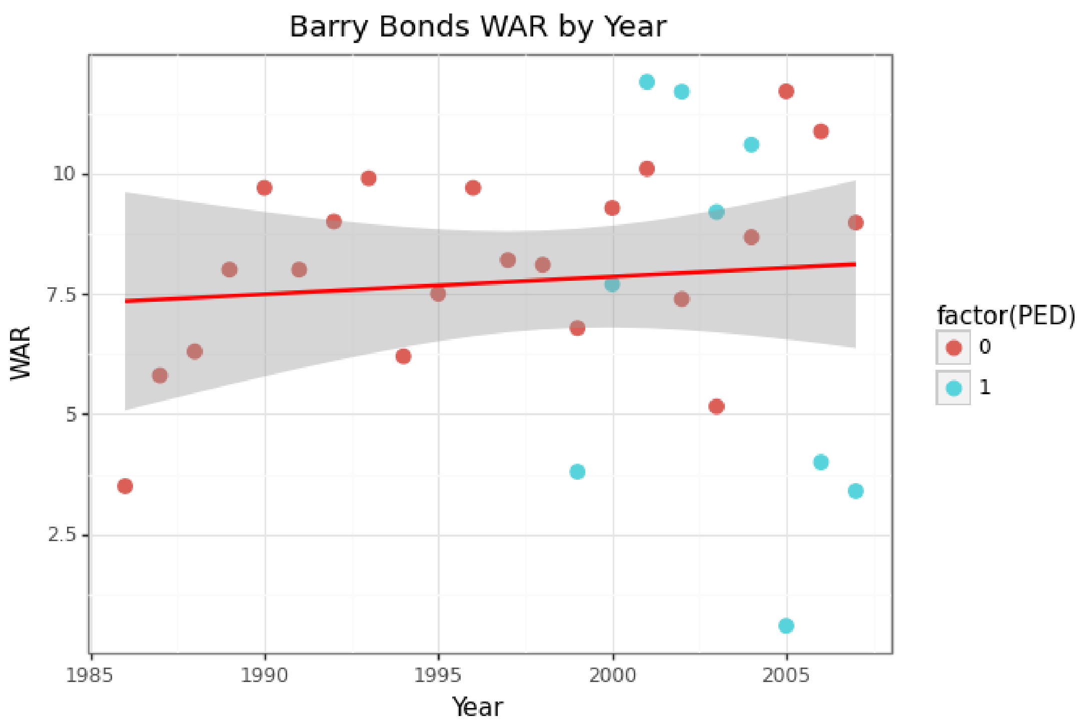
[68]:
```
# Calculating "accuracy" from predicted/expected WAR without PEDs to actual WAR␣
↪with PEDs
```

```
mse = mean_squared_error(post_PED_df['WAR'], predicted_non_PED_df['WAR'])
print("Mean Squared 'Error': ", mse)
```

```
Mean Squared 'Error':  28.337285941128496
```

[69]:

```
# Drawing a graph to represent Barry Bonds' WAR by year
(
    ggplot(combined_df)
    + aes(x = "Year", y = "WAR", color = "factor(PED)")
    + geom_point(size = 3)
    + geom_smooth(method = "lm", color = "red")
    # + geom_abline(slope = overall_slope, intercept = lr.intercept_, color =␣
 ↪'purple')
    + theme_bw()
    + labs(title = "Barry Bonds WAR by Year", x = "Year", y = "WAR")
)
```



[69]:

```
<ggplot: (8779879473552)>
```

[70]:

```
# Making another train test split to try and calculate home run totals
pre_PED_df = barry_bonds_df.loc[barry_bonds_df['PED'] == 0]
post_PED_df = barry_bonds_df.loc[barry_bonds_df['PED'] == 1]
```

```
predictors = ['OBP', 'BA', 'OPS', 'SLG', 'BB', '2B']
X = pre_PED_df[predictors]
y = pre_PED_df['HR']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

# refitting the StandardScaler with our new data
z.fit(X_train)

X_train[predictors] = z.fit_transform(X_train[predictors])
X_test[predictors] = z.transform(X_test[predictors])

# refitting the linear regression model
lr.fit(X_train, y_train)

# z-scoring PED stats
post_PED_df[predictors] = z.fit_transform(post_PED_df[predictors])
pre_PED_df[predictors] = z.fit_transform(pre_PED_df[predictors])
# lr.predict(pre_PED_df[predictors])[-9:]
```

[71]:
```
# Making the combined dataframe for home runs

combined_df = pd.DataFrame(barry_bonds_df[['Year', 'HR', 'PED']])
predicted_non_PED_df = pd.DataFrame({'Year': post_PED_df['Year']})
predicted_non_PED_df['HR'] = np.round(lr.predict(pre_PED_df[predictors])[-9:])
predicted_non_PED_df['PED'] = 0
combined_df = combined_df.append(predicted_non_PED_df)
print('Appended')
```

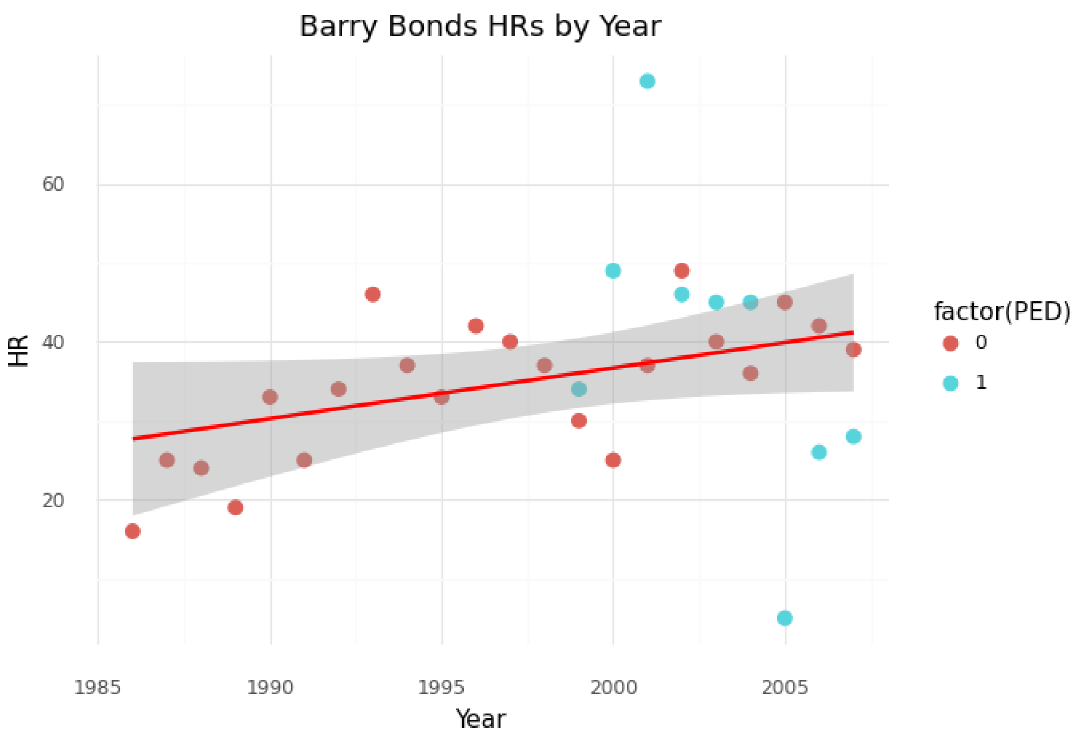
```
Appended
```

[72]:
```
# Finding the overall slope of the linear regression

coefs = lr.coef_
r_squared = lr.score(X, y)

# calculate overall slope
overall_slope = np.average(coefs, weights=abs(coefs)/sum(abs(coefs)))
overall_slope
lr.intercept_
```

[72]:
```
32.89999999999995
```

[73]:
```
# Drawing a graph to represent Barry Bonds' HRs by year
(
    ggplot(combined_df)
    + aes(x = "Year", y = "HR", color = "factor(PED)")
```

```
    + geom_point(size = 3)
    + geom_smooth(method = "lm", color = "red")
    # + geom_abline(slope = overall_slope, intercept = lr.intercept_, color =␣
 ↪'purple')
    + theme_minimal()
    + labs(title = "Barry Bonds HRs by Year", x = "Year", y = "HR")
)
```

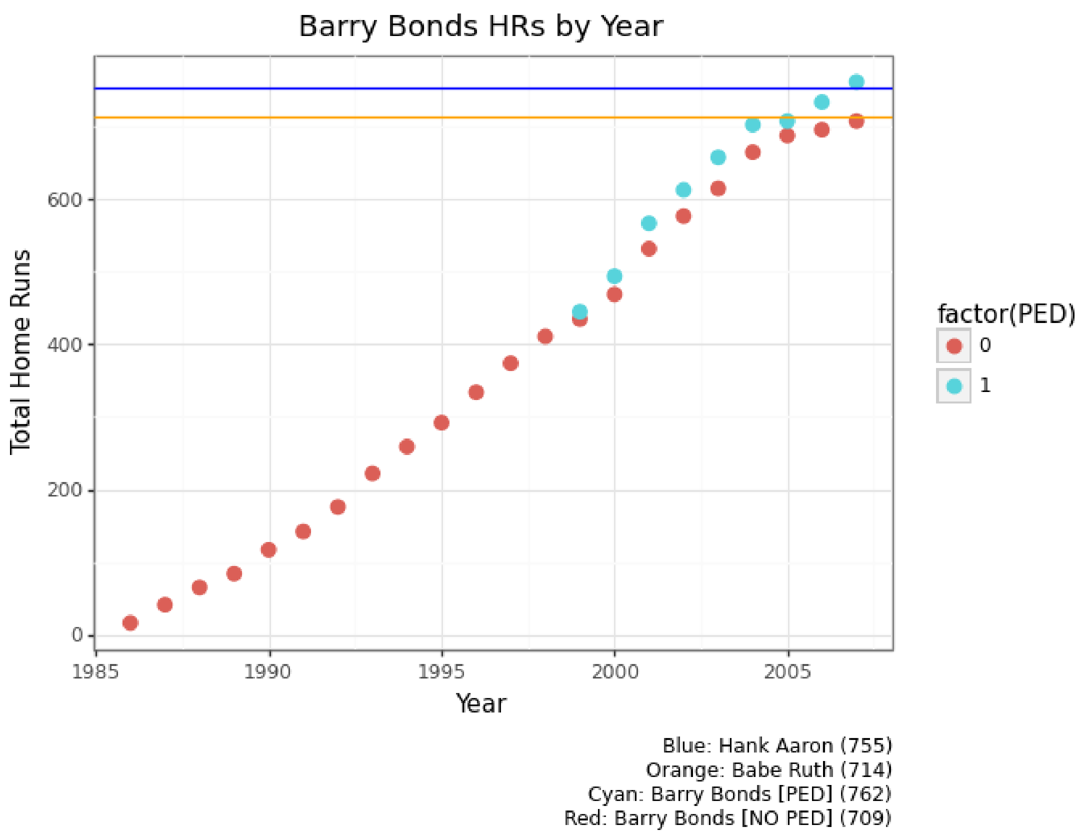


[73]: <ggplot: (8779878666553)>

[74]:
```
# Creating a dataframe of Barry Bonds' total home run count, with and without␣
 ↪PEDs

totalHR = pd.DataFrame(pre_PED_df[['Year', 'HR', 'PED']])
predicted_non_PED_df = pd.DataFrame({'Year': post_PED_df['Year']})
predicted_non_PED_df['HR'] = np.round(lr.predict(post_PED_df[predictors]))
predicted_non_PED_df['PED'] = 0
totalHR = totalHR.append(predicted_non_PED_df)
totalHR['total_HR'] = totalHR['HR'].cumsum()

totalHR_PED = pd.DataFrame(barry_bonds_df[['Year', 'HR', 'PED']])
totalHR_PED['total_HR'] = totalHR_PED['HR'].cumsum()
totalHR = totalHR.append(totalHR_PED.loc[totalHR_PED['PED'] == 1])
```

[75]:

```
# Drawing a graph to represent Barry Bonds' cumulative HRs by year
(
    ggplot(totalHR)
    + aes(x = "Year", y = "total_HR", color = "factor(PED)")
    + geom_point(size = 3)
    # + geom_smooth(method = "lm", color = "red")
    # + geom_abline(slope = overall_slope, intercept = lr.intercept_, color =
 ↪'purple')
    + geom_hline(yintercept = 755, color = 'blue')
    + geom_hline(yintercept = 714, color = 'orange')
    # + geom_text(aes(label='Hank Aaron (755)'), nudge_y=0.5))
    + theme_bw()
    + labs(title = "Barry Bonds HRs by Year", x = "Year", y = "Total Home
 ↪Runs", caption = "Blue: Hank Aaron (755)\nOrange: Babe Ruth (714)\nCyan:
 ↪Barry Bonds [PED] (762)\nRed: Barry Bonds [NO PED] (709)")
)
```



[75]: <ggplot: (8779879439496)>

0.1 Question 1 - Answer

Barry Bonds holds the pro-baseball record for most career home runs, however after the 1998 season, he started using illegal performance enhancing drugs (PEDs). How much of an impact did PEDs have on Barry Bonds' career stat totals? How would Barry Bonds' WAR and HR total look like if he had played for the same amount of time, but did not use PEDs?

By the 1998 season, Barry Bonds was unquestionably ranked as one of the best hitters in the history of baseball. Despite his accomplishments, much of the recognition and praise went towards Sammy Sosa and Mark McGwire, both of whom were using steroids and smashing records. Starting the 1999 season, Bonds started using PEDs himself out of jealousy to attempt to try and beat Sosa and McGwire.

Barry Bonds was a sure Hall of Fame player before he started doping. After using steroids, his power skyrocketed. He obtained the records he had sought after for so long, including the illusive all time career home run leader, but at the cost of his professional integrity.

What would Barry Bonds' career look like had he not used PEDs?

My first instinct was to predict Barry Bonds' WAR or "Wins Above Replacement", since it's the most complete statistic to measure a player's worth. To find this, I used the following predictors: - OBP (On-Base Percentage) - BA (Batting Average) - OBS (On-Base Plus Slugging) - SLG (Slugging) - BB (Walks) - 2B (Doubles) - HR (Home Runs)

I started by dividing Bonds' career stats into seasons before and after he started using PEDs. I created a 80/20 train-test split on the pre-PED dataset, z-scored the train and test statistics, then trained a linear regression model on the resulting train set.

Next, I created a dataframe with WAR, Years, and PED to hold all data points (predicted and real). Barry Bonds played 9 more seasons after doping so I predicted 9 more years of WAR based on his standard batting from the previous 9 seasons. I added that to the complete dataframe and graphed it.

Through this graph, we can see the huge difference in Bonds' WAR with PEDs compared to without PEDs. I calculated a mean squared error comparing how accurate (or rather inaccurate) the predicted WAR was compared to the actual steroids-enhanced WAR and ended up with a value of ~26.77, which is a huge difference. Just looking at the 2001 season alone shows the discrepancy, as Bonds accumulated a ridiculous 11.9 WAR, compared to the 9.4 he would have gotten without PEDs.

Our of curiosity, I also wanted to calculate Bonds' home run statistics with and without PEDs, since that was the record he was chasing the hardest. I followed the same steps as before, but this time predicting home runs. As expected, Bonds performed far better with PEDs. Something that the predicted values do not account for is him dipping down at the end of his career simply due to age slowing him down.

The most interesting graph in this question is the cumulative career total home runs statistic, which Barry Bonds currently holds at 762. According to the predictions from my model, Bonds would land just short of the great Babe Ruth in total home runs at the end of his career with 709, making him hypothetically the third most of all time.

0.2 Question 2: Clustering Via Age & Experience

When considering age, number of years played, Salary, BA, OPS, and WAR, what clusters form? What characterizes those clusters? Does player salary translate into better stats? - Variables Involved: Age (Interval), Years_Played (Interval), Salary (Interval), BA (Ratio), OPS (Continuous), WAR (Continuous) - Background: We will be measuring the career stat totals of players. Each of these variables are either interval (age, years_played, etc), or a percentage so we can more accurately measure older and newer players against each other. - Cleaning: We may drop outlier players who have not played enough games or were injured and didn't play. - Modeling/Computation: Use K-Means to generate a cluster of Players based on their age, years_played, Salary, etc, and observe how the generated clusters are characterized. - Graphs: Use a point graph to plot the players and visualize the clusters.

[76]:
```
batters_df = pd.read_csv('https://raw.githubusercontent.com/Jackmpauly/
 ↪CPSC-392-Final-Project/main/csv/batters_2022.csv')
batters_df.dropna(inplace = True)

batters_df['Salary'] = batters_df['Salary'].str.replace('$', '').str.
 ↪replace(',', '').astype(int)

batters_df.head()
```

[76]:
```
  last_name first_name     full_name  player_id  year  years_played  \
0     Judge      Aaron   Aaron Judge     592450  2022             7
1   Frazier       Adam  Adam Frazier     624428  2022             7
6   Bregman       Alex  Alex Bregman     608324  2022             7
7   Verdugo       Alex  Alex Verdugo     657077  2022             6
8   Rosario       Amed  Amed Rosario     642708  2022             6

   debut year  player_age  b_ab  b_total_pa  …  WAA  Rrep  RAR   WAR  \
0        2016          30   570         696  …  8.4    23  104  10.6
1        2016          30   541         602  … -1.3    21    9   0.7
6        2016          28   548         656  …  2.3    22   45   4.5
7        2017          26   593         644  … -1.0    22   13   1.1
8        2017          26   637         670  …  2.0    22   42   4.2

   waaWL%  162WL%  oWAR  dWAR  oRAR    Salary
0   0.554   0.552  10.4   0.0   101  19000000
1   0.492   0.492   0.7   0.3     8   8000000
6   0.515   0.515   5.0   0.0    49  13000000
7   0.494   0.494   1.7  -1.2    18   3550000
8   0.514   0.513   3.8   1.2    37   4950000

[5 rows x 40 columns]
```

[77]:
```
# make the predictors
```

```
# predictors = ['years_played', 'player_age', 'batting_avg',␣
↪'on_base_plus_slg', 'WAR', 'b_home_run', 'b_double']
# predictors = ['batting_avg', 'on_base_plus_slg', 'WAR', 'b_home_run',␣
↪'b_double']
# predictors = ['batting_avg', 'on_base_plus_slg', 'WAR', 'b_home_run',␣
↪'Salary']
predictors = ['years_played', 'player_age', 'batting_avg', 'on_base_plus_slg',␣
↪'WAR', 'b_home_run', 'Salary']

X = batters_df[predictors]

z = StandardScaler()
X[predictors] = z.fit_transform(X)

# NUM CLUSTERS
n = 3

km = KMeans(n_clusters = n)

km.fit(X)

print(silhouette_score(X, km.predict(X)))

X["clusters"] = km.predict(X)
batters_df["clusters"] = X["clusters"]
```

```
0.27480136770317914
```

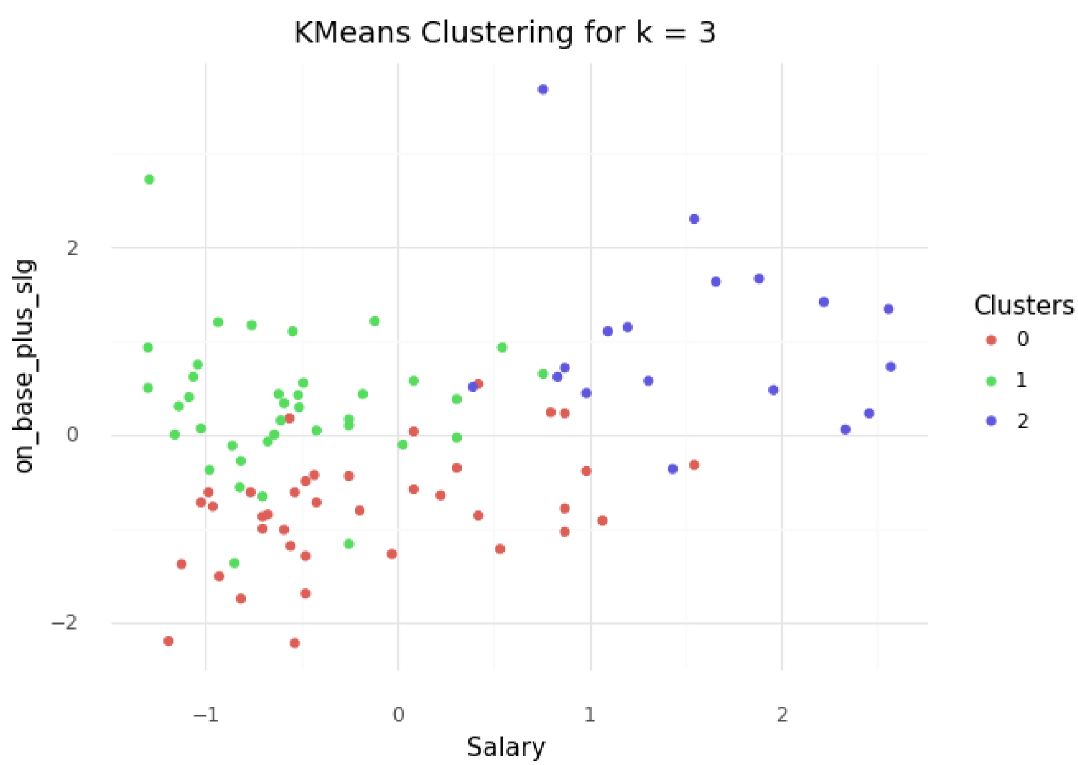
[78]:

```
# plot clusters function
def clusterPlot(df, x, y):
    plot = (
        ggplot(X)
        + aes(x = x, y = y, color = 'factor(clusters)')
        + geom_point()
        + theme_minimal()
        + labs(x = x, y = y, title = ("KMeans Clustering for k = 3"), color =␣
↪"Clusters")
    )
    print(plot)
```

[79]:

```
# plotting the clusters

# clusterPlot(X, 'b_double', 'WAR')
clusterPlot(batters_df, 'Salary', 'WAR')
clusterPlot(batters_df, 'Salary', 'on_base_plus_slg')
clusterPlot(batters_df, 'Salary', 'years_played')
```


KMeans Clustering for k = 3
WAR
Salary
Clusters
0
1
2
2
0
−2
−1
0
1
2


KMeans Clustering for k = 3
on_base_plus_slg
Salary
Clusters
0
1
2
2
0
−2
−1
0
1
2

0.3 Question 2

When considering age, number of years played, Salary, BA, OPS, and WAR, what clusters form? What characterizes those clusters? Does player salary translate into better stats?

Defining clusters was going to be tough, since many if not all standard batting stats are related to each other (eg: HRs is in part used to calculate OPS). A more interesting stat to cluster these statistics with was player value, specifically salary. Because salary is something that's predetermined before the season and unaffected by how well a player performs that year, it's interesting to see if a higher salary players performed better. In other words, does money motivate?

I created a KMeans cluster with n_clusters = 4. I used the following predictors: - years_played - player_age - batting_avg - on_base_plus_slg - WAR - b_home_run - Salary

That cluster produced a silhouette score of 0.542; not bad but also not good, but also to be expected with variables as varied as ours.

When plotting the clusters, three graphs particularly stuck out: Salary vs. WAR, Salary vs. OPS, and Salary vs. Years Played. In Salary vs. WAR, you can see the KMeans model create four clusters, where increasingly higher-paid clusters have a higher floor for how much WAR a player has accumulated. This makes sense as WAR and Salary are both technically measures of a player's worth; with one being economic value and the other being a calculated statistical value. I wanted to see if the clusters were has distinct for Salary vs. a standard batting statistic like OPS, and when I graphed it, similar clusters formed. The last plot I wanted to check out of curiousity was Salary vs. Years Played. I wanted to see if on average, more experienced players were paid more. This plot showed similar clusters, however there were more defined concentrated areas of points in the two clusters with lower salaries. This also made sense to me as teams usually follow a standard for paying newer players.

0.4 Question 3: Variables contributing to Batter WAR

When comparing a model using PCA on the R variables (rows 25-33) that contribute to WAR and retaining enough PCs to keep 90% of the variance, to a model using all other variables that contribute to WAR (rows 9-24) that contribute to WAR and retaining enough PCs to keep 90% of the variance, how much of a difference is there in the accuracy when predicting WAR? - Question was changed because the previous question didn't make much sense in in general and this question is more detailed. - Variables Involved: b_ab (Interval), b_total_pa (Interval), b_total_hits (Interval), b_single (Interval), b_double (Interval), b_triple (Interval), b_home_run (Interval), b_strikeout (Interval), b_walk (Interval), b_k_percent (Continuous), b_bb_percent (Continuous), batting_avg (Continuous), slg_percent (Continuous), on_base_percent (Continuous), on_base_plus_slg (Continuous), woba (Continuous), Rbat (Continuous), Rbr (Continuous), Rdp (Continuous), Rdef (Continuous), Rpos (Continuous), RAA (Interval), WAA (Continuous), Rrep (Interval), RAR (Interval) - Cleaning: We only used players who had the minimum plate appearances to qualify for the batting title (502 or more PA). - Modeling/Computation: Use PCA and use the values it gives us to see which variables are the most important then use Linear Regression to see which stat had the most impact - Graphs: Two graphs that plot cumulative variance vs. # of PCs with a y-intercept at .9.

[ ]:
```
batters_df = pd.read_csv('https://raw.githubusercontent.com/Jackmpauly/
 ↪CPSC-392-Final-Project/main/csv/batters_2022.csv')
batters_df.head()
```

[ ]:
```
  last_name first_name      full_name  player_id  year  years_played  \
0     Judge      Aaron    Aaron Judge     592450  2022             7
1   Frazier       Adam   Adam Frazier     624428  2022             7
2    Garcia     Adolis  Adolis Garcia     666969  2022             5
3   Pollock         AJ     AJ Pollock     572041  2022            11
4      Bohm       Alec      Alec Bohm     664761  2022             3

   debut year  player_age  b_ab  b_total_pa  …  WAA  Rrep  RAR   WAR  \
0        2016          30   570         696  …  8.4    23  104  10.6
1        2016          30   541         602  … -1.3    21    9   0.7
2        2018          29   605         657  …  1.4    22   36   3.6
3        2012          34   489         527  … -1.3    18    6   0.4
4        2020          25   586         631  … -1.2    22    9   0.8

   waaWL%  162WL%  oWAR  dWAR  oRAR       Salary
0   0.554   0.552  10.4   0.0   101  $19,000,000
```

```
1   0.492   0.492   0.7   0.3     8   $8,000,000
2   0.509   0.509   3.2   0.0    32          NaN
3   0.491   0.492   0.5  -0.4     6          NaN
4   0.492   0.492   2.6  -1.5    26          NaN

[5 rows x 40 columns]
```

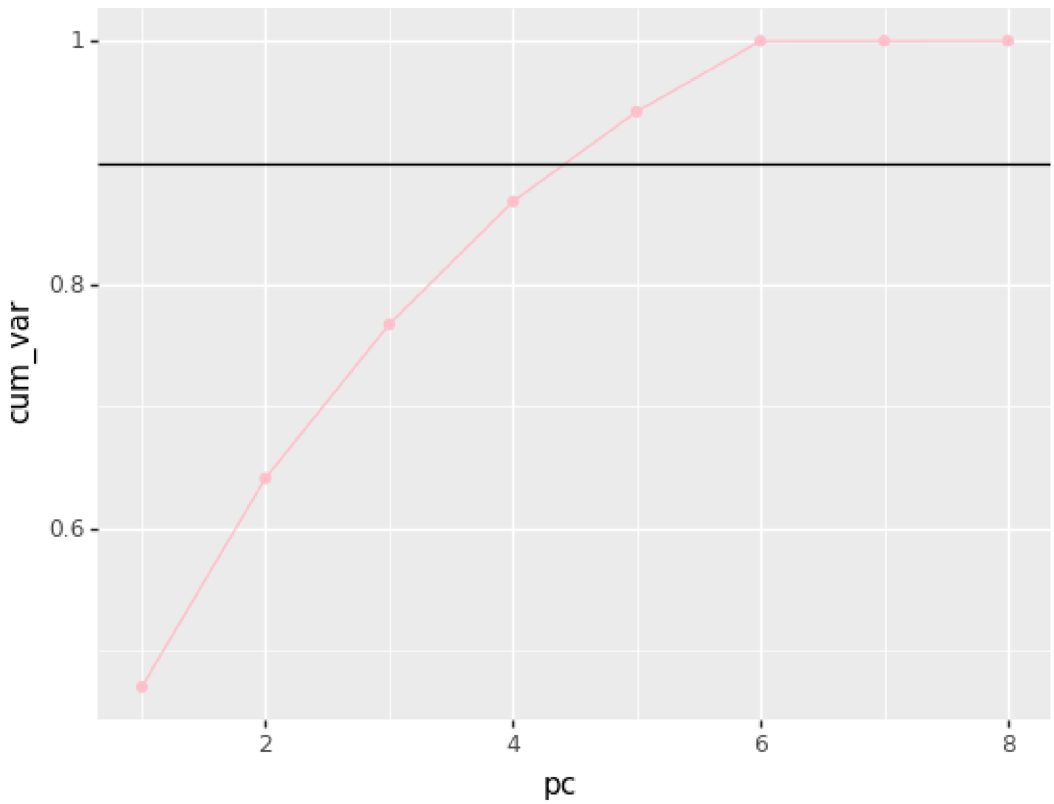
#R variables (Rows 25-33)

[ ]:
```
features = batters_df.columns[25:33]
z = StandardScaler()
batters_df[features] = z.fit_transform(batters_df[features])

pca = PCA()
pca.fit(batters_df[features])

pcaDF = pd.DataFrame({"expl_var" :
                      pca.explained_variance_ratio_,
                      "pc": range(1,9),
                      "cum_var":
                      pca.explained_variance_ratio_.cumsum()})
pcaDF.head()
```

[ ]:

	expl_var	pc	cum_var
0	0.469511	1	0.469511
1	0.171487	2	0.640998
2	0.126323	3	0.767322
3	0.100762	4	0.868084
4	0.073853	5	0.941937

[ ]:
```
#pc vs cumulative variance

(ggplot(pcaDF, aes(x = "pc", y = "cum_var")) + geom_line(color = "pink") +
 geom_point(color = "pink") + geom_hline(yintercept = 0.90))
```



[ ]: <ggplot: (8786180478577)>

[ ]:
```
pcomps5 = pca.transform(batters_df[features])
pcomps5 = pd.DataFrame(pcomps5[:, 0:5])

#all data for R variables
lr = LinearRegression()
lr.fit(batters_df[features], batters_df["WAR"])
print("all data: ", lr.score(batters_df[features], batters_df["WAR"]))

#5 PCs for R variables
lr2 = LinearRegression()
lr2.fit(pcomps5, batters_df["WAR"])
print("5 PCs:    ", lr2.score(pcomps5, batters_df["WAR"]))
```

```
all data:  0.9992449914284373
5 PCs:     0.9846599356530585
```

#All Other Variables that contribute to WAR (Rows 9-24)

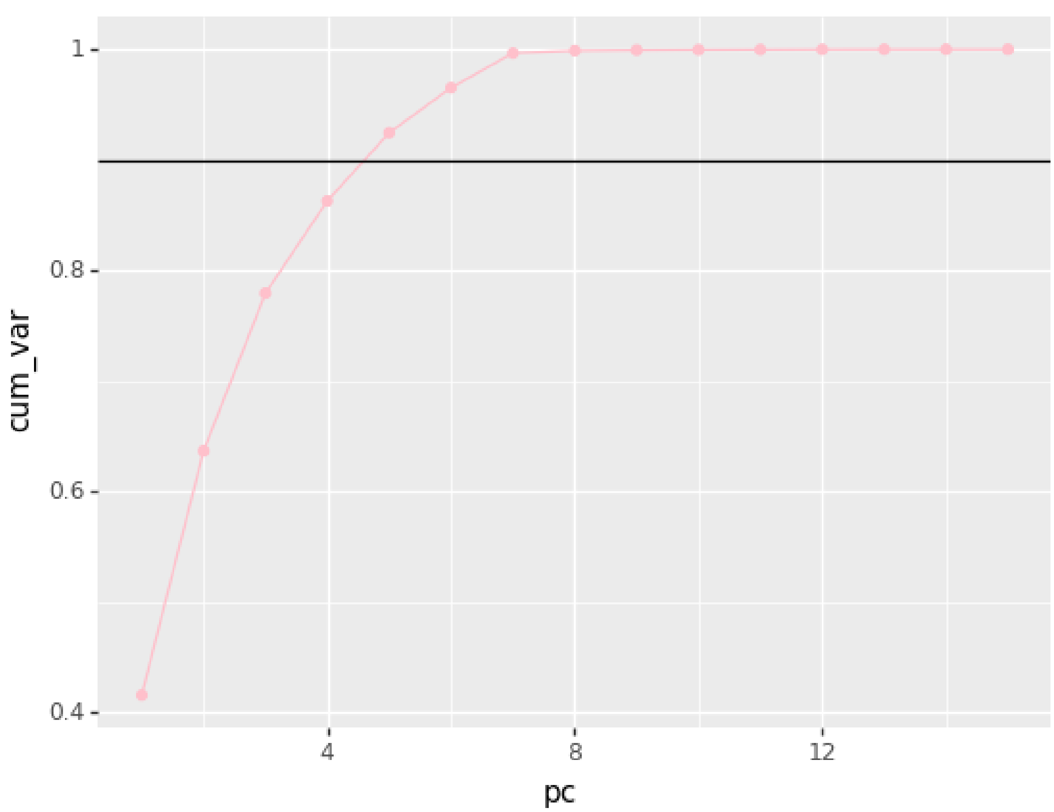
[ ]:
```
features2 = batters_df.columns[9:24]
z = StandardScaler()
```

```python
batters_df[features2] = z.fit_transform(batters_df[features2])

pca2 = PCA()
pca2.fit(batters_df[features2])

pcaDF2 = pd.DataFrame({"expl_var" :
                      pca2.explained_variance_ratio_,
                      "pc": range(1,16),
                      "cum_var":
                      pca2.explained_variance_ratio_.cumsum()})
pcaDF2.head()
```

[ ]:

	expl_var	pc	cum_var
0	0.415936	1	0.415936
1	0.220938	2	0.636874
2	0.142586	3	0.779460
3	0.083500	4	0.862960
4	0.061558	5	0.924517

[ ]:
```python
#pc vs cumulative variance
(ggplot(pcaDF2, aes(x = "pc", y = "cum_var")) + geom_line(color = "pink") +
 geom_point(color = "pink") + geom_hline(yintercept = 0.90))
```



[ ]: <ggplot: (8786113784337)>

[ ]:
```
pcomps5 = pca2.transform(batters_df[features2])
pcomps5 = pd.DataFrame(pcomps5[:, 0:5])

#all data for other variables
lr3 = LinearRegression()
lr3.fit(batters_df[features2], batters_df["WAR"])
print("all data: ", lr3.score(batters_df[features2], batters_df["WAR"]))

#5 PCs for R variables
lr4 = LinearRegression()
lr4.fit(pcomps5, batters_df["WAR"])
print("5 PCs:    ", lr4.score(pcomps5, batters_df["WAR"]))
```

```
all data:  0.6055433081037255
5 PCs:     0.5800392397564103
```

##Q3 Answer

WAR, or wins above replacement, is an advanced baseball statistic that aims to measure a baseball player's value based on all aspects of their game, which for hitters would include skills such as hitting, fielding and baserunning. The stat is based on a replacement level player who is an imaginary player for that specific season who is replacable (not very good). The two most accepted forms of WAR are bWAR and fWAR, or Baseball Reference WAR and FanGraphs WAR. They are separate since they are calculated using different variables. For our models we look at bWAR since all of our data is from Baseball Reference, and since that is now clear, we will refer to bWAR as just WAR.

When looking at solely the R variables that contribute to WAR, the accuracy for all of the data is very close to 1, with taking in all R variables leading to an accuracy of about .999, and using only 5 PCs gave an accuracy of .985. This is a stark contrast from the accuracys when taking into account all other variables. When using the rest of the variables, the accuracy is about .606 and when only taking into account 5 PCs, the accuracy is .580. Since the accuracy gets slightly decreased when only using 5 PCs in both models it can be inferred that the model's performance gets slightly worse when performing PCA on these models. However, another conclusion that we can derive from these findings is that the R variables are much more important to the calculaton of WAR compared to all other variables. This is exhibited by the overall higher accuracys in the R variable models. This would make sense as the R variables are much more advanced statistics that are calculated using the other variables. Since WAR is calculated using advanced metrics, it would make sense that the variables that are more elementary contribute less in a player's WAR.

0.5 Question 4: Pitcher FIP & ERA discrepancy

What pitchers had the highest discrepancy between their FIP and ERA? Look at their team's defensive runs saved to see if the defense had anything to do with it. - Question was changed to add complexity and utilize fielding stats somewhere in our analysis. - Variables Involved: FIP (Continuous), ERA (Continuous), Tm_rDRS (Interval), H9 (Continuous), HR9 (Continuous) -

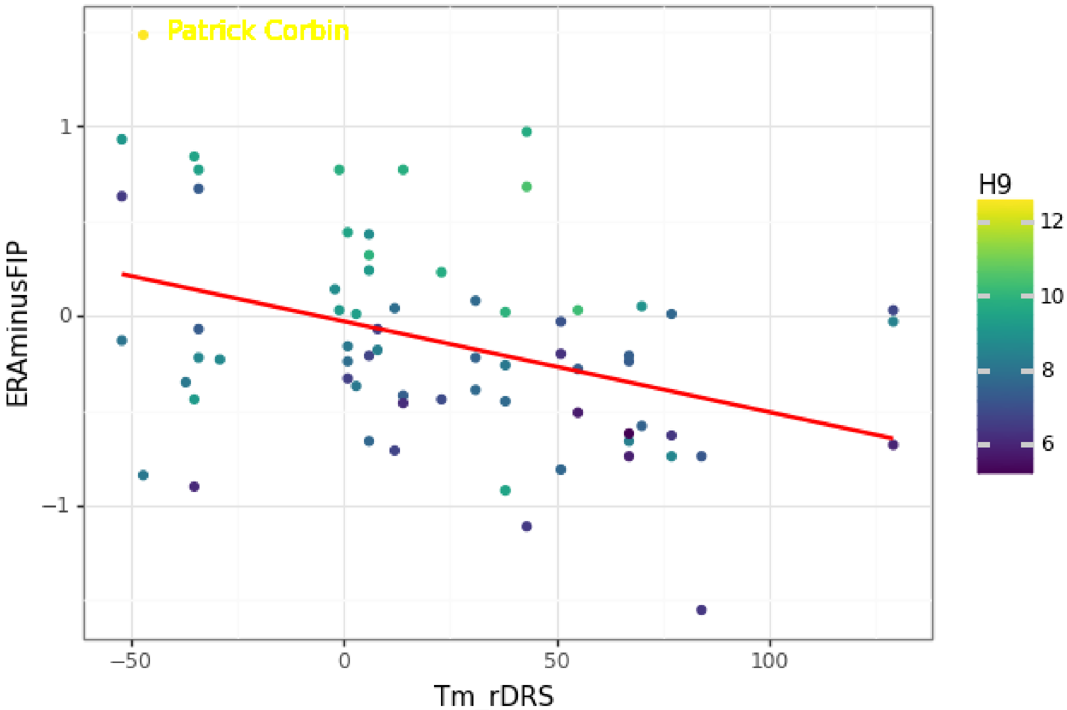
Cleaning: We will only be looking at pitchers who qualify for the ERA title (pitched at least 162 innings) as ERA and FIP is a stat that can be taken out of context without enough innings. - Modeling/Computation: Create a graph where team DRS is the x variable and ERA - FIP is the y variable to see if there happens to be a trend between them. Use a linear regression model to see if there is any sort of relationship. - Graphs: Two graphs that are plotted as Tm_rDRS vs ERA - FIP and are color coded based on H9 (hits per nine innings) and HR9 (home runs per nine innings).

[ ]:
```
pitchers_df = pd.read_csv('https://raw.githubusercontent.com/Jackmpauly/
↪CPSC-392-Final-Project/main/csv/pitchers_2022.csv')
pitchers_df.head()
```

[ ]:
```
    Rk               Name  Age   Tm  Tm_rDRS  Lg   W  L   W-L%   ERA  …   BF  \
0  806  Justin Verlander   39  HOU     67.0  AL  18  4  0.818  1.75  …  666
1  858      Kyle Wright   26  ATL     31.0  NL  21  5  0.808  3.19  …  738
2  855  Brandon Woodruff   29  MIL     51.0  NL  13  4  0.765  3.05  …  620
3  627     Cal Quantrill   27  CLE     77.0  AL  15  5  0.750  3.38  …  770
4  265        Zac Gallen   26  ARI     55.0  NL  12  4  0.750  2.54  …  714

   ERA+   FIP   WHIP   H9  HR9  BB9   SO9  SO/W Name-additional
0   223  2.49  0.829  6.0  0.6  1.5   9.5  6.38       verlaju01
1   128  3.58  1.159  7.8  0.9  2.6   8.7  3.28       wrighky01
2   129  3.08  1.070  7.2  1.1  2.5  11.2  4.52       woodrbr01
3   113  4.12  1.208  8.6  1.0  2.3   6.2  2.72       quantca01
4   158  3.05  0.913  5.9  0.7  2.3   9.4  4.09       galleza01

[5 rows x 37 columns]
```

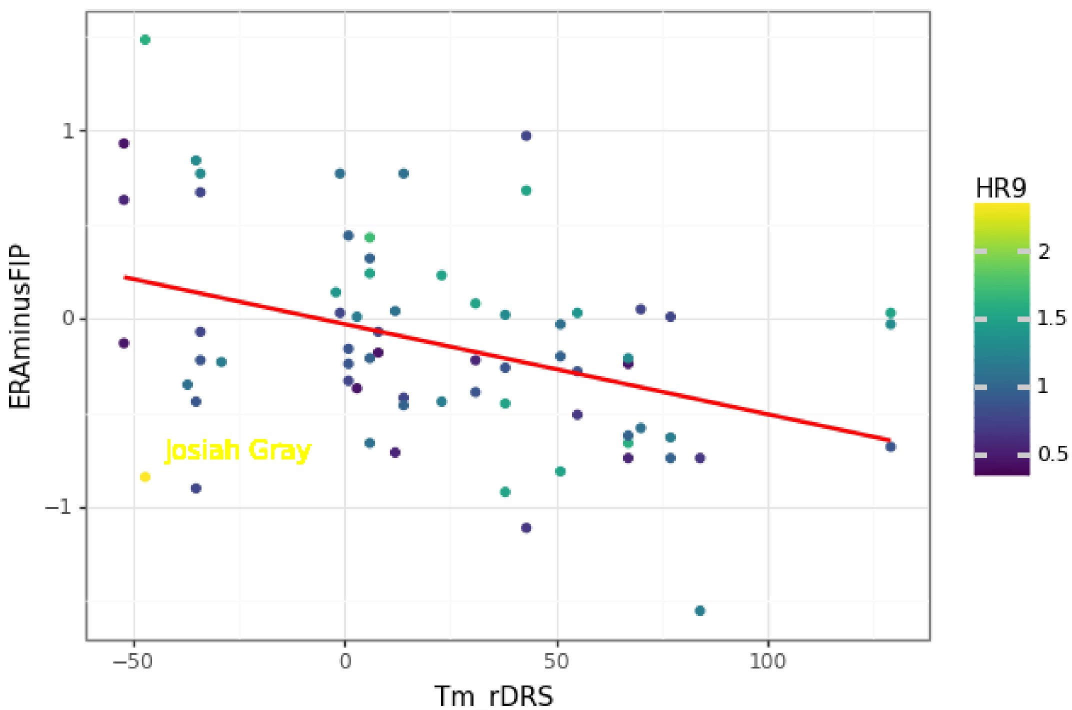
[ ]:
```
#create a new column in the dataframe of ERA minus FIP
pitchers_df['ERAminusFIP'] = pitchers_df['ERA'] - pitchers_df['FIP']
```

[ ]:
```
ggplot(pitchers_df, aes(x = "Tm_rDRS", y = "ERAminusFIP", color='H9')) +␣
↪geom_point() + theme_bw() + geom_smooth(method='lm',  se=False, color =␣
↪"red") + geom_text(label = "Patrick Corbin", x = -20, y = 1.5, color =␣
↪"yellow")
```



[ ]: <ggplot: (8786113807075)>

```
[ ]: ggplot(pitchers_df, aes(x = "Tm_rDRS", y = "ERAminusFIP", color='HR9')) +␣
     ↪geom_point() + theme_bw() + geom_smooth(method='lm',  se=False, color =␣
     ↪"red") + geom_text(label = "Josiah Gray", x = -25, y = -.7, color = "yellow")
```



[ ]: <ggplot: (8786113649613)>

##Q4 Answer

ERA is a stat that has been the most commonly used to evaluate a pitcher, but its most glaring flaw is that it usually doesn't account for the defense behind the pitcher. On the otherhand, FIP has been created to counteract this problem by only taking into account outcomes that the pitcher can control which includes home runs, walks, and strikeouts. TM_rDRS is a stat that measures how many runs a team has saved by playing good defense. Since FIP eliminated defense from the equation, taking the difference of FIP from ERA would tell us whether a pitcher has been getting bailed out by their defense or not. A pitcher getting bailed out by their defense would be something along the lines of a fielder making a great play to get an out and prevent a run from scoring. If the result of ERA - FIP is > 0, that usually means that the pitcher is performing better than what their ERA indicates and that their defense may be letting them down. If the result of ERA - FIP is < 0, that usually means that the pitcher is performing worse than what their ERA indicates and that their defense may be bailing them out.

In general there is a slight negative relationship between ERA - FIP and Tm_rDRS which makes sense as the better the defense gets, the more a pitcher would get bailed out by their defense. The outliers in this dataset can be explained through futher anaylsis.

In my two examples above I have plotted Tm_rDRS against ERA - FIP and would like to analyze two pitchers, Corbin and Gray, who were both on the Nationals who had abysmal -47 Tm_rDRS in 2022. If defense is truly a factor in the discrepancy between FIP and ERA, why does Corbin have a high ERA - FIP and Gray have a low ERA - FIP? To answer this question I colored the points based on how many hits they gave up per nine innings (H9) and how many home runs they gave up per nine innings (HR9).

In Corbin's case, he has a high H9 and a high ERA - FIP. How can this be if I mentioned earlier that he would be overperforming his ERA? The key here is that the Nationals' defense is bad. This would entail that Corbin is giving up a lot of hits because the defense is not able to cover enough ground to make hard catches or make errors on balls that could have been fielded by better fielders.

In Gray's case, he has a high HR9 and a low ERA - FIP. This would suggest that he is doing poorly but getting bailed out by his defense; however we established earlier that the Nationals' defense is terrible. The key here is that Gray gives up a large amount of home runs compared to other pitchers which shows why he is doing so poorly if his defense is doing well and bailing him out. The defense has no impact on home runs since the ball travels out play.

1 Convert to PDF

[ ]:
```
# doesn't show this cells output when downloading PDF
!pip install gwpy &> /dev/null

# installing necessary files
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!sudo apt-get update
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic

# installing pypandoc
!pip install pypandoc

# connecting your google drive
from google.colab import drive
drive.mount('/content/drive')

# copying your file over. Change "Class6-Completed.ipynb" to whatever your file␣
↪is called (see top of notebook)
!cp "drive/My Drive/Colab Notebooks/Pauly, Inn - Baseball Final Project.ipynb" .
↪/

# Again, replace "Class6-Completed.ipynb" to whatever your file is called (see␣
↪top of notebook)
!jupyter nbconvert --to PDF "Pauly, Inn - Baseball Final Project.ipynb"
```

```
Reading package lists… Done
Building dependency tree
Reading state information… Done
pandoc is already the newest version (2.5-3build2).
texlive is already the newest version (2019.20200218-1).
texlive-latex-extra is already the newest version (2019.202000218-1).
texlive-xetex is already the newest version (2019.20200218-1).
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
```

```
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:2 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu focal InRelease
Hit:3 http://ppa.launchpad.net/cran/libgit2/ubuntu focal InRelease
Hit:4 https://cloud.r-project.org/bin/linux/ubuntu focal-cran40/ InRelease
Hit:5 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu focal InRelease
Hit:6 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64
InRelease
Hit:7 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu focal InRelease
Hit:8 http://ppa.launchpad.net/ubuntugis/ppa/ubuntu focal InRelease
Hit:9 http://archive.ubuntu.com/ubuntu focal InRelease
Get:10 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Fetched 336 kB in 1s (228 kB/s)
Reading package lists… Done
Reading package lists… Done
Building dependency tree
Reading state information… Done
texlive-fonts-recommended is already the newest version (2019.20200218-1).
texlive-plain-generic is already the newest version (2019.202000218-1).
texlive-xetex is already the newest version (2019.20200218-1).
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
```